Teknisk dokumentation

Redaktör: Hanna Nyqvist

Version 0.1

Status

Granskad		
Godkänd		

PROJEKTIDENTITET

2010/VT, Grupp 3
Linköpings Tekniska Högskola

Gruppdeltagare

Namn	**Ansvar**	**Telefon**	**E-post**
Joakim Gebart	Projektledare (PL)	073-065 13 83	joage546@student.liu.se
Hanna Nyqvist	Dokumentansvarig (DOK)	070-617 07 24	hanny589@student.liu.se
Jonatan Olofsson	Systemansvarig färdplan, kommunikation (SAP)	076-321 91 05	jonol402@student.liu.se
Sara Örn	Systemansvarig framdrivning (SAF)	070-350 73 17	saror792@student.liu.se
Erik Levholt	Systemansvarig sensorer (SAS)	070-662 09 52	erile330@student.liu.se
Christoffer Peters	Systemansvarig datorprogram (SAD)	070-170 69 55	chrpe822@student.liu.se
Marcus Eriksson	Systemansvarig logik (SAL)	073-647 71 80	marer014@student.liu.se

E-postlista för hela gruppen: tsea27@jge.se
Hemsida: http://webcollab.solidba.se

Kund: ISY, Linköpings universitet, 581 83 Linköping
Kontaktperson hos kund: Lennart Bengtsson, 013-28 13 67, lennartb@isy.liu.se
Kursansvarig: Thomas Svensson, 013-28 13 68, thomass@isy.liu.se
Handledare: Andreas Ehliar, 013-28 89 56, ehliar@isy.liu.se

Innehåll

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2010-05-17	Första utkast.		

1 Inledning

Dokumentet beskriver detaljerat den färdiga kartrobot som konstruerats. Kartroboten har byggts av en grupp på sju personer inom ramen för kursen TSEA27, Elektronikprojekt Y.

1.1 Bakgrund och syfte

Projektet har utförts inom ramen för kursen TSEA27 Elektronikprojekt Y. Ett syfte är att genomföra ett projekt enligt LIPS-modellen, för att få god träning inför projekt i arbetslivet. Ett annat syfte är att knyta ihop kunskaper från olika kurser genom att bygga och programmera en robot som ska kartlägga en bana.

1.2 Källor

Roboten har byggts enligt beskrivningarna i dokumenten *Designspecifikation*[1], *Projektdirektiv*[4] och *Kravspecifikation*[3].

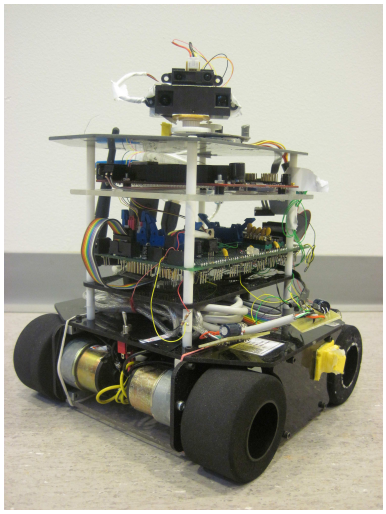
1.3 Skrivkonventioner

Med datorn avses den bärbara dator som roboten kopplas till, och datorprogrammet är det program som körs på den.

uint8_t och sint8_t är egendefinierade typer, anpassade för de AVR-processorer som används. uint8_t är ett 8-bitars heltal utan tecken och sint8_t är ett 8-bitars heltal med tecken.

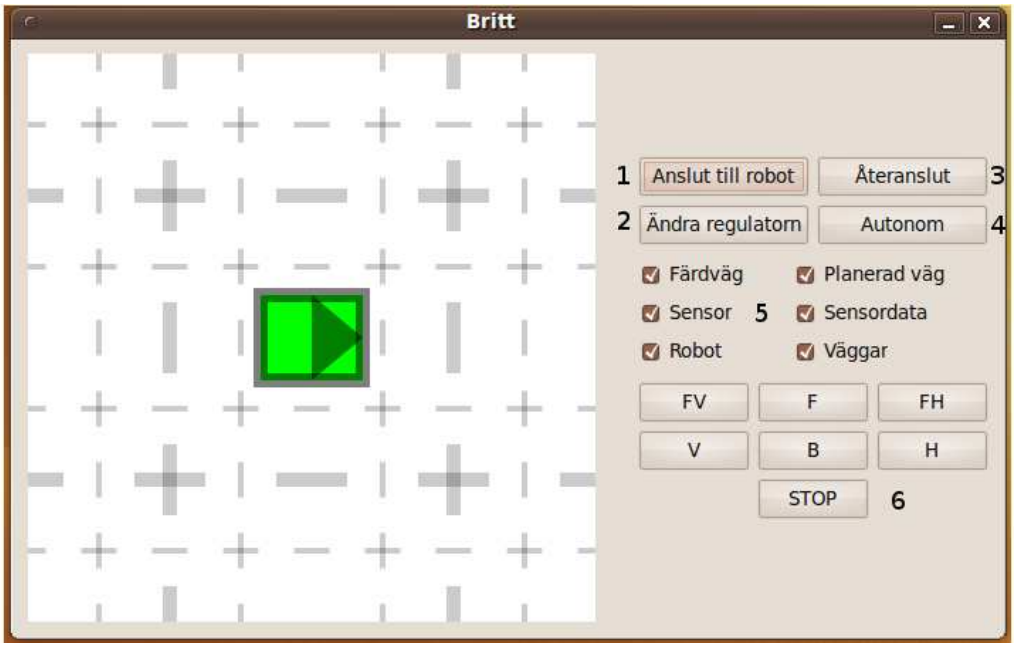
2 Produkten

Boten Berit är en kartritande robot , se bild 1, som autonomt kör igenom en bana bestående av 80 cm långa kartongsegment ihopsatta enligt banspecifikationen definierad i *Kravspecifikation*[3]. Roboten kan även fjärrstyras.



Figur 1: Roboten *Boten Berit*

Roboten styrs med hjälp av ett datorprogram på en bärbar dator. Då datorprogrammet startas kommer vyn i figur 2 upp.



Figur 2: Startvy för datorprogram.

För att starta systemet skapas ett trådlöst nätverk med namnet *B-rit* för kommunikation mellan datorprogrammet och roboten. Roboten startas sedan genom att koppla in ett 7.2-voltsbatteri och slå på strömomkopplaren på roboten. Roboten ansluter själv till nätverket med namnet *B-rit* och ger sig själv IP-numret 192.168.192.168. Efter att roboten startats kan man i datorprogrammet ansluta trådlöst till roboten genom att trycka på knapp *1 Anslut till robot* och ange den ovan nämnda IP-adressen samt portnummer 6715, vilket visas i figur 3.

Figur 3: Vy då användaren vill ansluta till roboten.

Då systemet startats befinner sig roboten i manuellt läge. Roboten kan då fjärrstyras med hjälp av *6 styrknapparna* i datorprogrammet. För att byta körläge till autonomt trycker man på knappen *4 Autonom*. Man kan även byta tillbaka till manuellt läge genom att trycka på samma knapp igen, denna har nu bytt text till *4 Manuell*.

I autonomt körläge kan användaren inte påverka robotens rörelser förutom att roboten stannar då man trycker på *6 STOP*. Användaren kan dock klicka i de olika *5 kryssrutorna* som finns för att se olika data som beräknas eller mäts av roboten medan den kör. I vyn kommer robotens interna karta att ritas upp. Väggarna tonas efter hur säker roboten är på att de existerar, ju starkare färg desto mer säkert är det att de finns. Rödtonade väggar är väggar som roboten ännu inte anser finns men ändå har fått några mätvärden på. Blåtonade väggar däremot är väggar som med säkerhet fastställts existerar. Magentafärgade väggar är väggar som roboten passerat igenom.

I autonomt läge kan användaren ändra de regulatorparametrar som används av styrmodulen genom att trycka på knapp *2 Ändra regulatorn*. Detta påverkar robotens körbeteende och om dessa ändras kan det inte längre garanteras att roboten kör säkert och kan kartlägga banan.

Om anslutningen till roboten går ner kan man lätt återansluta genom att klicka på *3 Återanslut*. Då används den IP-adress och det portnummer som angavs då datorprogrammet första gången anslöts mot roboten.

3 Teori

I roboten används olika algoritmer. De två som beskrivs här är de mest komplicerade. Övriga, som till exempel PD-regulatorn och minstakvadratanpasssning, är mer standardmässiga.

3.1 Mätvärden

När mätpunkter inkommer till logikdelen kommer denna att filtrera bort punkter som ligger nära en skärning av rutnätslinjerna, vilket gör mätningen tvetydig pga mätbrus, eller om den ligger mitt i en ruta, vilket den aldrig borde kunna göra eftersom alla väggar ska ligga på rutnätslinjerna. De punkter som överlever denna filtrering kommer att approximeras till närmaste vägg.

Varje vägg ges poäng för inkommna mätpunkter som träffar den och ges poängavdrag för mätningar som ser igenom den. En jämförelsesats för hur stor poäng en vägg behöver för att faktiskt räknas som en vägg av planeringsdelen finns definierad som $WALL\_PREDICATE$ i map.hpp.

Efter att roboten har kört igenom en vägg och empiriskt bevisat att ingen vägg finns där sätts det speciella värdet 255 som poäng. Detta betyder att det absolut aldrig är en vägg där även om några brusiga mätpunkter råkar hamna på den väggen. I en vidareutveckling av projektet skulle kalibreringsalgoritmen kunna ta hänsyn till dessa väggar och räkna bort dessa för att minska risken för felkalibrering, se 3.3.2.

3.2 Vägplanering

Roboten använder en egenutvecklad algoritm för att planera en väg till nästa outforskade kartområde. Algoritmen letar inledningsvis upp de hörn som finns i robotens inre kartrepresentation. Samtidigt sparas de hörnpunkter som inte har två anslutande ändar undan som "lösa ändar", och bland dessa väljs sedan den punkt ut som ligger närmast roboten. Utav fyra alternativa punkter, en i varje väderstreck, väljs återigen den punkt som ligger närmast roboten. Denna används som målpunkt.

Runt alla hörn som hittats placeras sedan åtta punkter symmetriskt. De punkter som på grund av att de ligger på eller för nära en vägg slängs, medan resten används för att hitta den kortaste giltiga vägen till målpunkten.

3.3 SLAM

Roboten använder sig av en SLAM-algoritm - Simultaneous Localisation And Mapping - för att korrigera sin position och vinkel i kartan. Algoritmen utnyttjar flera mätpunkter på inlästa väggar och använder minsta-kvadratberäkningar för att bestämma robotens drift. Kalibrering görs under färd, men algoritmen ställer också upp villkor för när roboten skall stanna för att inhämta fler mätvärden för att noggrannare kunna justera in sig i kartan.

SLAM-algoritmen utnyttjas också initialt för att finjustera grundkalibreringen vid uppstart av autonomt läge.

De två kalibreringsmetoderna utnyttjar samma matematik men algoritmerna skiljer sig något åt beroende på om roboten är i kalibreringsläge, då den parkerar och ökar sensorsvepvidden till 360^o. Nedan beskrivs de två algoritmerna mer detaljerat.

3.3.1 Löpande SLAM

Löpande SLAM är den mindre komplicerade algoritmen av de två och det är ett krav att man förstår denna för att förstå beskrivningen av kalibrerläget. Samtliga konstanter som anges i stycket nedan finns definierade i Logic.hpp.

När en mätpunkt överlevt filtreringen kommer den uppmätta punkten att sparas i en separat SLAM-buffert tillhörande den vägg som den approximeras till. När $SLAM\_buffer\_count\_limit$ mätpunkter har hamnat i bufferten och minst $SLAM\_min\_time$ tid har förflutit kommer den löpande SLAM-algoritmen att köras på de mätpunkter som finns lagrade i bufferten.

Samtliga punkter i bufferten approximeras genom minstakvadratmetoden till en linje. Om linjen är minst $\sqrt{SLAM\_line\_min\_length\_squared}$ lång och inte mer än $SLAM\_angle\_correction\_limit$ radianer avvikelse från den förväntade orienteringen på väggen kommer robotens vinkel och position att justeras med $vinkelavvikelsen \cdot SLAM\_angle\_factor$ respektive $positionsfelet \cdot SLAM\_position\_factor$.

Konstanterna i denna justering är valda så att den ska inträffa ofta under körning och bara göra väldigt små ändringar till robotens position och vinkel, dock kommer ändå ett större fel att inträffa vid större rotationer, speciellt under stillastående pga slirande däck och imperfektioner i verkliga världen. För att åtgärda detta krävs en omkalibrering, denna beskrivs i 3.3.2.

3.3.2 Kalibrerläge

Vid kalibrering parkeras motorerna och sensortornet ställs att svepa 360^o. Logikdelen tar in $NR\_CALIBRATION\_VALUES$ st mätpunkter (definition i Logic.hpp, i skrivande stund definierad till 480 st vilket motsvarar 2 varv på tornet) som lagras i en separat kalibrerbuffert utan att sättas in i väggarna.

Efter insamlingen är färdig ansätts en position och vinkel och samtliga mätpunkter läggs in i den SLAM-buffert som hör till den närmaste väggen, med samma filtrering som vid vanlig insättning i vägg, se 3.3.1 för en beskrivning av SLAM-buffertar. När hela kalibrerbufferten har filtrerats ut eller satts in i SLAM-buffertar görs för varje buffert en minstakvadratanpassning av samtliga punkter i den. Medelkvadratfelet samt vinkelavvikelsen och antalet mätpunkter beräknas ihop till en poängsats för den ansatta positionen och vinkeln.

Detta upprepas för ett antal olika positioner och vinklar för att testa fram vilken kombination som passar bäst in på robotens verkliga position. Den position och vinkel med minsta felen kommer att antas vara korrekt och robotens position och vinkel sätts till dessa.

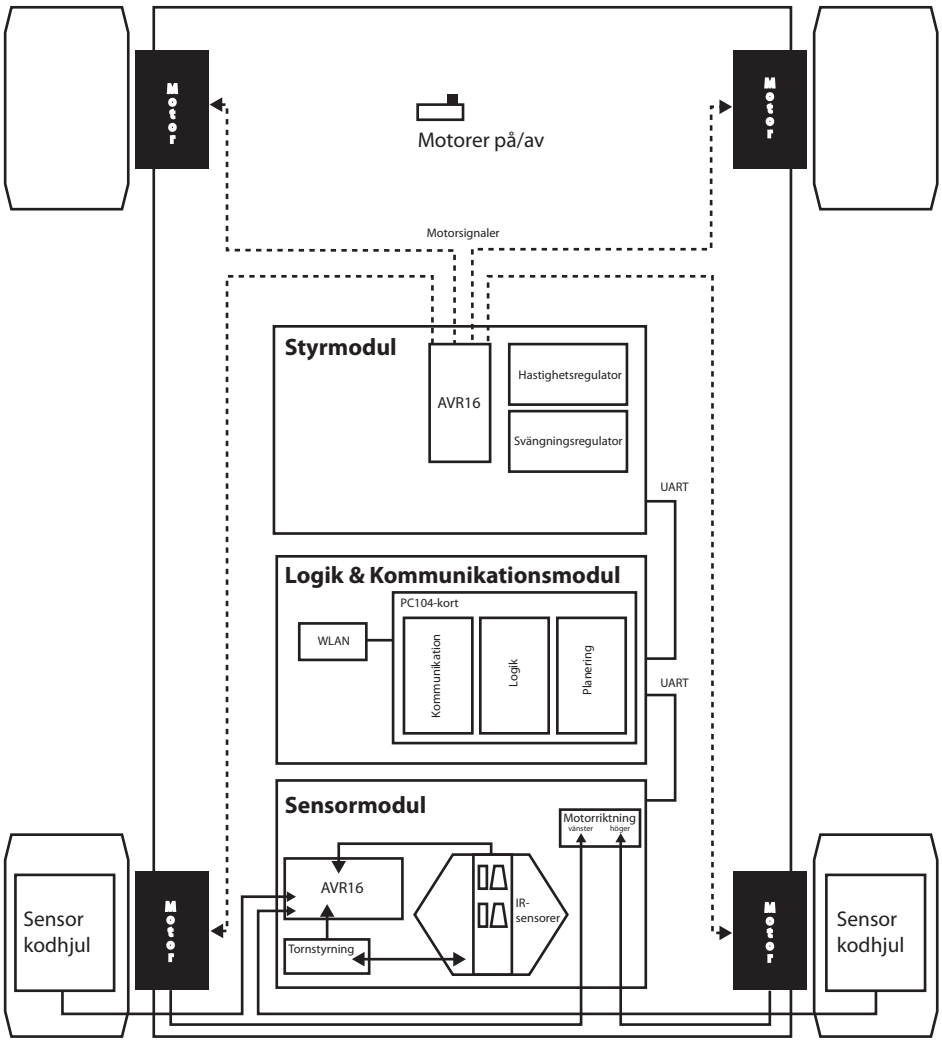
När en position och vinkel har uppdaterats sätts hela kalibrerbufferten in i kartan genom $insert\_in\_map()$ på samma sätt som när vanliga mätpunkter inkommer, se 3.1.

4 Översikt av systemet

Roboten har två körinställningar, autonom och manuell styrning. I autonomt läge letar sig roboten på egen hand igenom en bana som uppfyller i förväg bestämda krav och samlar in data om banans utseende. Denna data skickas sedan trådlöst till en dator med ett datorprogram som presenterar resultatet, en karta över banan. I manuellt läge fjärrstyrs roboten med hjälp av datorprogrammet. I detta läge samlas dock ingen kartdata in.

Systemet består av två huvudkomponenter, en robot och en bärbar dator med WLAN. Roboten hämtar kartdata och sänder dessa via WLAN till datorn som ritar upp en karta över den bana roboten befinner sig i. Roboten kan även fjärrstyras från datorn. För att underlätta felsökningen kan även information som sensordata och reglerparametrar med mera sändas till datorn och presenteras där.

Roboten i sin tur är indelad i flera moduler, var och en med en tydlig uppgift. Modulerna är sensormodul, logikmodul, styrmodul samt kommunikationsmodul och alla dessa är utbytbara och uppgraderbara. Moduler som körs på separata processorer kommunicerar med varandra via serieportar enligt blockschemat, bild 4.



Figur 4: Systemskiss över robotens moduluppdelning

Sensor- och styrmodulen består av var sin AVR-processor, Atmega16, medan kommunikations- och logikmodulen ligger som var sin mjukvarumodul på en mer komplex ARM-processor.

5 Intermodulär kommunikation

All kommunikation mellan robotens olika moduler sker via kommunikationsmodulen. Kommunikationen mellan de av robotens systemmoduler som arbetar på olika processorer sker via seriell anslutning (RS-232). Detta innebär att all kommunikation sker via kommunikationsmodulen. Styr- och sensormodulerna har var sin seriell anslutning till kommunikationsmodulen, via vilken all kommunikation går. Detta gör att de kan begära eller skicka data på eget initiativ, utan att behöva ta hänsyn till om den andra modulen sänder något för tillfället.

Meddelanden mellan moduler delas alltid upp i fyra delar

Datafält:	Funktion	Mål	Ursprung	Datalängd	Data
Antal bitar:	8	4	4	8	(bestäms av datalängd men är begränsad till 8 bytes)

Mål- och ursprungsfälten kan lämnas tomma vid direktkommunikation mellan modulen och dess serieportsadapter i kommunikationsmodulen, och används främst då en modul begär data från en annan modul via kommunikationsmodulen, samt vid datorns kommunikation med kommunikationsmodulen.

Vid funktions- eller dataförfrågan innehåller den första delen av meddelandet information om vilken typ av funktion som efterfrågas av mottagarmodulen - till exempel uppdatering av parameter eller sändande av data. Då data efterfrågas skickas funktionen *GIMMEH* tillsammans med ett globalt nummer som representerar den datatyp som efterfrågas.

Då data sänds från modulen innehåller det första fältet istället det specifika globala nummer på datat som specificerats - samma nummer som har efterfrågats för att skapa dataförfrågan. Datat återfinns i datafältet.

Funktionsnumrena definieras global för alla moduler och representeras internt som konstanter med numeriska värden (ENUM). Ett antal funktioner är reserverade för globala funktioner, och resterande är modulspecifika.

I detta dokument användes följande syntax för att beskriva intermodulär kommunikation.

Källa	**Funktion**	**Data**	**Mål**
COMM→ STEER:	REQUEST_DATA	-	-
STEER:	GIMMEH	TARGET_ANGLE	LOGIC

Modulernas kommandon beskrivs närmare i dokumentet under respektive modul.

De gemensamma kommandon som skall kunna mottas av varje modul är:

Funktion	**Data**	**Beskrivning**
GIMMEH	variabelnummer	Begär data från modulen
STOP_GIMMEH	variabelnummer	Begär att modulen skall sluta skicka data av en viss typ
RESET		Starta om modulen
IDENTIFY		Begär en enstaka byte med den siffra som associerats med modulen

Då ett GIMMEH-kommando mottagits av en modul skickas den efterfrågade datan normalt kontinuerligt tills dess att ett STOP_GIMMEH-kommando erhålls.

6 Sensormodul

Sensormodulen är den modul som insamlar och bearbetar mätdata från de sensorer som finns på roboten. Modulen tar fram information om robotens omgivning som till exempel avstånd och vinklar till väggar. Den tar även fram information om robotens rörelse.

För att mäta avstånd till väggarna har två IR-sensorer med olika arbetsområden placerats på ett vridbart torn. Olika arbetsområden används för att kunna få bästa möjliga noggrannhet i olika intervall. De arbetsområden vi har på sensorerna är enligt databladen 10-80 cm för att se på nära håll och 40-300 cm för att se på längre håll. Praktiska mätningar har visat att sensorerna är pålitliga i intervallen så nära roboten man kan komma"(10 cm) - 50 cm för den korta, och från 30 cm - 150 cm för den långa. Det vridbara tornet kontrolleras med en stegmotor, som också

styrs av sensormodulen. Vinkeln till väggen kan beräknas med hjälp av tornets vridning på roboten tillsammans med robotens vridning i rummet.

För att styra tornets stegmotor har 4 ben på en ATmega16 kopplats till en ULN2803-krets som förstärker strömmarna så de räcker för att driva motorn. En unipolär stegmotor används, så den positiva matningen kopplas till den gemensamma polen på motorn, och de 4 ändpunkterna på motorns lindningar kopplas till de 4 utgångarna på ULN2803-kretsen. Mjukvaran skickar ut rätt sekvens av ettor och nollor på de fyra pinnarna, och håller dessutom reda på i vilken av sina 8 faser motorn befinner sig. För att få en mjukare gång används tekniken med halvsteg, och motorn går igenom sekvensen A, AB, B, BA-, A-, A-B-, B-, B-A för att sedan börja om igen med A. A- och B- är när motsvarande lindning som A och B är strömsatt i motsatt riktning.

Vi valde att delvis implementera hela sekvensen som ett stort steg, vilket gör mjukvaran betydligt mycket enklare men minskar noggrannheten till 1/8, Då utväxlingen som används med motorn gör att noggrannheten och upplösningen blir tillräcklig ändå. Då ser man till att alla stegsekvenser avslutar i tex läge A, så får steg åt höger alltid börja med AB och steg åt vänster alltid börja med B-A. Man måste även hålla rätt på antal steg som tagits, för att veta i vilken riktning tornet med sensorerna pekar relativt roboten. De underliggande rutinerna för stegmotorstyrningen kör dock med motorns fulla upplösning.

För att utföra en avståndsmätning ställs först AD-muxen in till rätt sensor, och en A/D-omvandling genomförs. När A/D-omvandlingen är klar anropas en funktion som slår i en tabell för att räkna om det erhållna 10 bitars binära talet till ett avstånd i cm. Om det aktuella mätvärdet ligger nära övre eller undre gränsen på den valda sensorns mätintervall görs nästa mätning med den andra sensorn. Sensormodulen försöker på detta vis välja rätt sensor beroende på resultatet från förra mätningen.

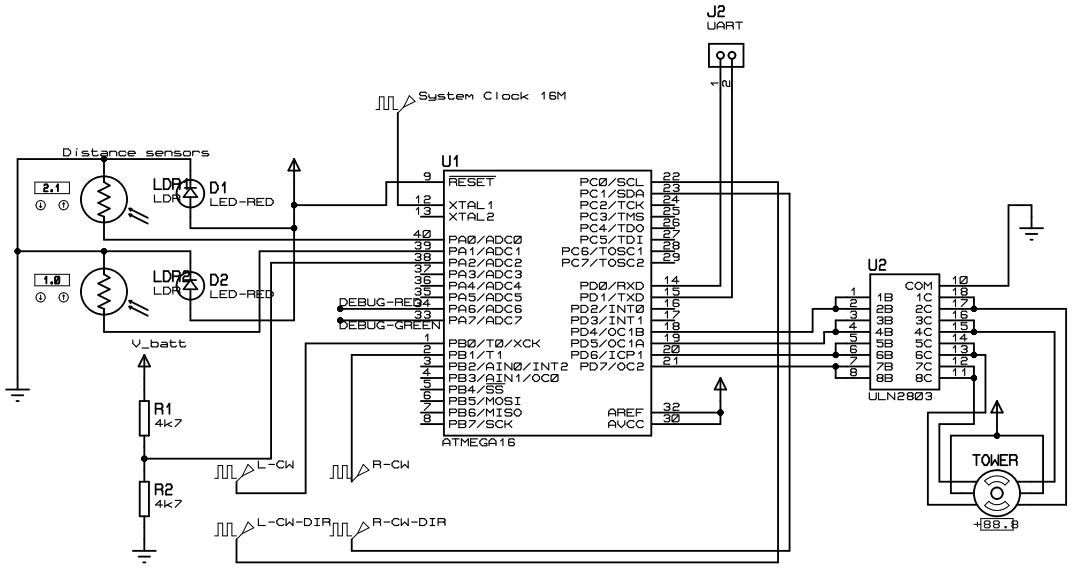
När avståndet mätts och omvandlats till cm lagras det tillsammans med den vinkel tornet stod i när mätningen gjordes och sändas vidare till logik/ruttplanering. När ett avståndsvärde skickas sänds avståndet först, följt av vinkeln, och beställaren av avståndsdata får själv räkna trigonometri för att avgöra var väggen befinner sig. Om avståndet som erhålls från sensorn med den längsta räckvidden är för långt för att det ska vara pålitligt skickas istället en nolla, för att det ska gå att använda denna information till att ta bort väggar som inte finns.

Sensormodulen kan också ta emot kommandon ifrån logikmodulen beträffande vilka vinklar den skall hålla sig inom vid avsökning, och hur stora steg den skall göra mellan varje mätpunkt.

För att få fram information om robotens rörelse i rummet samplas mätdata från kodhjul placerade på robotens hjul. Logikmodulen kan med hjälp av dessa data beräkna position samt vridning för roboten.

6.1 Kopplingsschema

Nedan i figur 5 visas ett kopplingsschema för sensormodulen.



Figur 5: Kopplingsschema för sensormodulen

6.2 Modulspecifika kommandon

Följande kommandon kan tas emot av sensormodulen från datorn för att ändra beteendet:

Funktion	Data	Beskrivning
SWEEP_ANGLES	int,int	Begränsa sensortornets rotation till att ligga mellan två vinklar
SWEEP_ANGLES_STEP	int	Ange hur tätt mätningar sker
SENSOR_RANGE	enum	Short/long/auto vilken avståndsmätare som ska användas

6.3 Tillgängliga data

Följande data går att begära från sensormodulen genom ett GIMMEH-kommando:

Datanamn	Typ	Innehåll
SWEEP_ANGLES	int,int	Vinklar som tornet sveper mellan
SWEEP_ANGLE_STEP	int	Vinkelsteg
RANGE_POINT	int,int	Avstånd till mätpunkt samt vinkel på tornet när mätningen utfördes
CODE_WHEEL_TICKS	int,int	Antal pulser från kodskivorna på hjulen. Ordning: höger, vänster

7 Logik- och kommunikationsmodul

Under denna rubrik beskrivs de mjukvarumoduler som tillsammans ligger på robotens pc104-kort. Kortet kör en egenbyggd Linuxkärna (version 2.6.33.2 med egna tillägg) som startar operativsystemet Gentoo Linux från en USB-minnessticka, WLAN-kommunikationen ansluts även den över USB. För felsökning är det möjligt att ansluta en seriell terminal (vt100 med RS-232 signaler) till kortet.

7.1 Logik

Logikmodulens uppgift är att hålla reda på vad roboten håller på med, och vad den ska göra härnäst.

Då roboten är i autonomt läge skall roboten inledningsvis kalibreras efter banans rutnät. Detta görs genom att begära mätningar ifrån sensormodulen i ett antal vinklar runtom roboten. Genom att jämföra de inkomna mätvärdena kan roboten med hjälp av minstakvadratberäkningar och SLAM-korrigeringar bestämma sin position och orientering i rummet. De inkomna mätpunkterna utgör sedan grunden i den interna kartrepresentationen för att påbörja en sökning efter bästa outforskade område att ta sig till.

Den interna kartan representeras som ett rutnät (en matris) av 80 cm långa möjliga väggbitar. Matrisens storlek begränsas av den i *Kravspecifikationen* förbestämda maximala banstorleken. För varje möjlig vägg i matrisen anges en sannolikhet för att just denna vägg består av en kartongvägg. Sannolikheten baseras på antalet mätpunkter på väggen, men också på om väggen har blivit genomlyst av andra mätningar, eller till och med tidigare korsad av roboten.

Logikmodulen tar in mätdata ifrån sensormodulen och approximerar dessa till punkter i ett rutnät som den använder för att rita upp en virtuell karta. Mätpunkterna approximeras till den närmsta möjliga väggen i den interna kartrepresentationen. Punkter som då hamnar nära en skärning mellan flera möjliga väggdelar filtreras dock bort eftersom det är svårt att bestämma viken av väggarna som denna punkt bör approximeras till. Detta innebär att bara mätpunkter som hamnat på de mittersta 40 centimetrarna av de 80 centimeter långa väggdelarna anses vara tillförlitliga mätvärden. Om avståndssensorerna mäter igenom en vägg straffas både den vägg som mätts upp och den vägg som setts igenom genom att sannolikheten för att dessa väggbitar är kartongväggar minskas. När en möjlig vägg har fått tillräckligt många mätvärden approximerade till sig anses väggen vara en kartongvägg.

Den interna karta som ritas upp används som grund för slutvillkorsberäkning och färdplanering.

Rummet anses kartlagt då alla upptäckta väggar ansluter till två andra väggar. Detta krav kan också appliceras på fall med en eventuell köksö.

Robotens rutt planeras genom att hitta de områden på den interna kartan som ännu inte utforskats (dvs de väggar som ännu inte har två anslutningar). Den närmsta av dessa punkter väljs som destination och en planeringsalgoritm beräknar utifrån den en väg att följa som undviker kända väggar. Se avsnitt *Planering* nedan. Då en vägg ändrar status under färden görs planeringen om och en nytt målpunkt beräknas baserat på den nya informationen. Planeringen görs också om om SLAM-algoritmen flyttar på roboten.

Logiken håller också reda på att SLAM-kalibrering sker tillräckligt ofta, baserat på olika faktorer som tid och vinkeländring sedan senaste kalibreringen.

Från kommunikationsmodulen kan logikmodulen i manuellt läge också ta emot fjärrstyrsignaler. Dessa kommandon behandlas och skickas direkt till styrmodulen, utan att passera färdplaneringen. I manuellt läge sker ingen kartritning och inga hinder detekteras eller undviks.

På begäran från kommunikationsmodulen kan logikmodulen skicka kartdata och planerade checkpoints som skickas vidare till datorn. Även utskrifter för SLAM-kalibrering och andra data kan skickas till datorprogrammet.

Planering
Planeringen tar in koordinater till den plats logikdelen bestämmer att roboten ska åka till. Den planerar utifrån detta en väg bestående av ett antal checkpoints som gör att roboten kan ta sig dit utan att krocka med de väggar som redan är kartlagda.

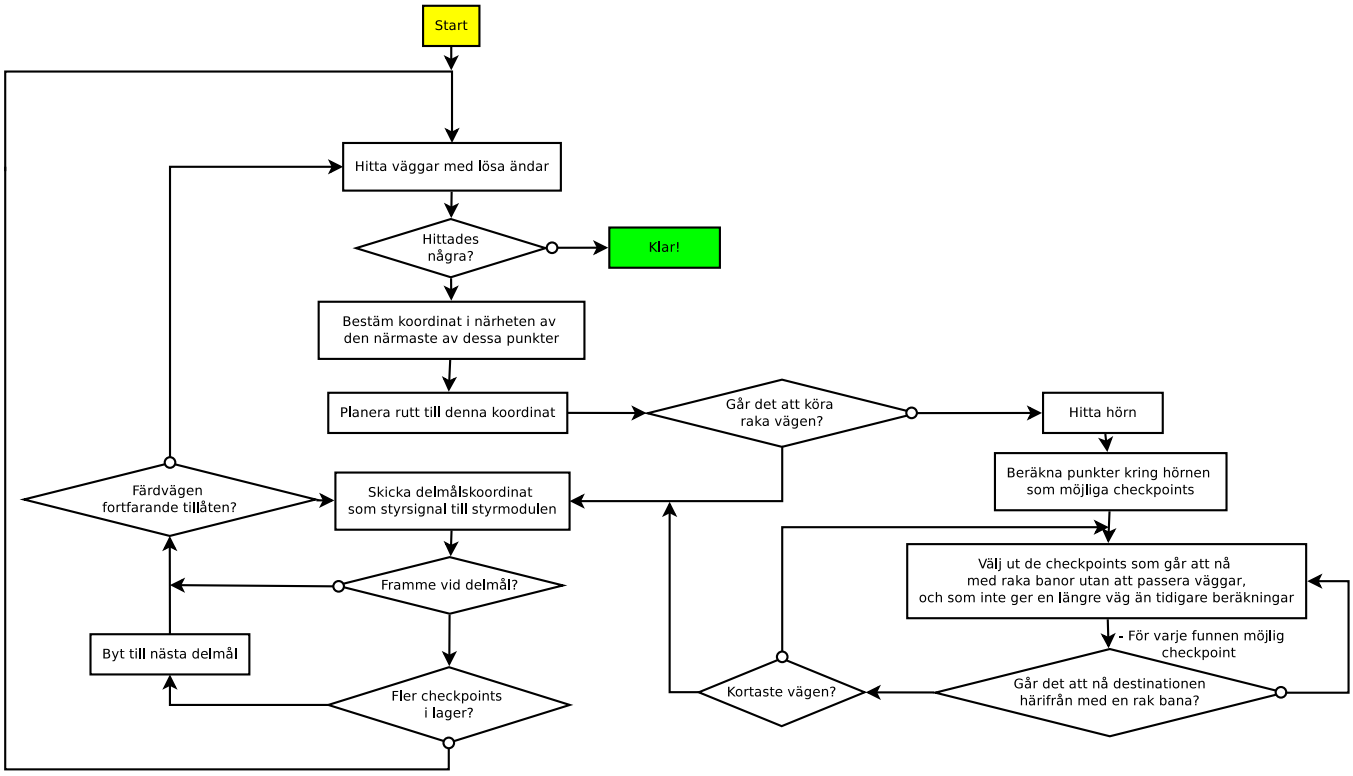
För att välja de checkpoints som ger kortast väg mellan robot och mål utgår algoritmen från potentiella checkpoints i närheten av detekterade hörn i kartan, och jämför möjliga vägar för att hitta den kortaste tillåtna vägen. Utifrån beräknade checkpoints levereras en relativ vinkel och ett avstånd till nästa checkpoint till framdrivningen som styr roboten till önskad punkt.

SLAM
Under körningen i autonomt läge körs även två SLAM-funktioner för att korrigera för den drivning av robotens position samt vridningsvinkel som uppstår på grund av att hjulen slirar.

SLAM-funktionen för att korrigera positionen fungerar på följande sätt. Det uppmätta avståndet till en vägg jämförs med det avstånd till samma vägg som logikmodulen beräknat att roboten har. Robotens position korrigeras sedan med en konstant del av skillnaden mellan dessa värden. Eftersom många mätvärden från varje vägg erhålls kommer den totala korrigeringen bli relativt rättvisande trots mätbrus.

7.2 Flödesschema

I bild 6 nedan visas ett flödesschema för logikmodulens beteende.



Figur 6: Flödesschema över logikmodulen

7.2.1 Modulspecifika kommandon

Följande kommandon kan tas emot av logikmodulen från datorn för att ändra beteendet:

Funktion	Data	Beskrivning
START_AUTONOMOUS_MODE	inget	Startar det autonoma programmet och inleder kalibrering
STOP_AUTONOMOUS_MODE	inget	Avbryter det autonoma programmet
STEER_COMMAND	data	Skickar en önskad positionsförändring för manuell styrning

7.2.2 Tillgängliga data

Följande data går att begära från logikmodulen genom ett GIMMEH-kommando:

Datanamn	Typ	Innehåll
CURRENT_MAP	int	Den uppmätta kartan
CHECKPOINTS	int	Planerade checkpoints
CURRENT_POSITION	float,float	Robotens nuvarande position
CURRENT_ANGLE	float	Robotens uppmätta vinkel

7.3 Kommunikation

Kommunikationsmodulen är en mjukvarumodul som hanterar intern kommunikation och anslutningen till datorn via WLAN.

Modulen tar emot kommandon från datorn som innehåller information om vilken modul som är mottagare, vilken modulfunktion som efterfrågas, samt data till funktionen. Den inkomna datan skickas sedan till mottagarmodulen genom robotens interna kommunikationsgränssnitt.

Kommunikationsmodulen ansvarar för att fråga modulerna efter data som efterfrågats av andra moduler och vidarebefordra denna.

Då en modul fått ett vidarebefodrat meddelande från kommunikationsmodulen med funktionen GIMMEH ska den efterfrågade datan sändas kontinuerligt tills dess att ett nytt meddelande med funktionen STOP_GIMMEH erhållits.

7.3.1 Modulspecifika kommandon

Följande kommandon kan tas emot av komm-modulen för att ändra beteendet:

Funktion	Data	Beskrivning
COLLECT_DATA	int	Skapar ett antal meddelanden från datorn som alltid ska skickas

7.3.2 Tillgängliga data

Kommunikationsmodulen tillhandahåller inga data som kan läsas genom GIMMEH.

8 Styrmodul

Styrmodulen är den modul som styr de motorer som kontrollerar robotens hjul och ansvarar därmed för att roboten färdas till den position som beräknats av Logik-modulen. Roboten har två separata motorer som styr hjulen på var sin sida om roboten oberoende av varandra.

8.1 Regulator och återkoppling

Med hjälp av en PD-regulator reglerar styrmodulen robotens hastighet och rotation. Från Logikmodulens färdplanering tar styrmodulen emot en målkoordinat att föra roboten till. Denna uttrycks i polära koordinater relativt roboten. Styrmodulen mottar alltså den relativa vinkeln mellan robotens färdriktning och riktningen till målet samt det relativa avståndet från roboten till målet. En positiv vinkel innebär att målet ligger till vänster om roboten och rotation moturs kommer därmed att ske. En negativ vinkel innebär att målet ligger till höger om roboten.

Eftersom koordinaterna anges relativt roboten kan koordinaterna ses som felet i position, och regleringen kan därmed ske direkt utgående från koordinaterna. Därför behöver styrmodulen inte motta data om var roboten befinner sig i kartans koordinatsystem. De relativa koordinaterna måste uppdateras kontinuerligt i takt med att roboten rör sig, även om inte målpunkten flyttar sig i rummet (dvs. i banans koordinatsystem), detta för att regleralgoritmen ska kunna reglera med en tillfredsställande noggrannhet. Återkopplingen till styrmodulen från logikmodulen sker dessutom med fördel med jämna tidssteg eftersom styrmodulen inte själv håller ordning på hur lång tid som gått mellan samplingarna utan beräknar en ny styrsignal att lägga ut på motorerna så snart ny återkoppling erhållits.

PD-regulatorn beräknar en styrsignal enligt:

$\theta =$ relativ vinkel mellan robotens färdriktning och målet.
$\omega = \dot{\theta} =$ robotens rotationshastighet relativt målet.
$s =$ styrsignal

$$s = P \cdot \theta + D \cdot \omega$$

P och D är regulatorkonstanter som kalibrerats för att generera lagom kraftig styrsignal. Dessa kan dock justeras med hjälp av datorprogrammet. Initialt är $P = 1$ och $D = 3$.

Derivatan av θ, ω, approximeras genom att ta differensen mellan var 10:e sampelvärde av θ.

8.2 Pulsbreddsmodulerad signal till motorerna

Beroende på hur nära målet roboten är bestäms en grundhastighet hos motorerna. Följande grundhastigheter gäller:

Avstånd [cm]	Hastighet [hex]
> 50	0x90
<= 50 och > 20	0x60
<= 20	0x40

Styrsignalen som beräknats av PD-regulatorn enligt tidigare beskrivning adderas sedan med denna grundhastighet för att bestämma den signal som ska läggas på den ena motorn. Signalen som läggs på den andra motorn beräknas genom att istället subtrahera styrsignalen från grundhastigheten. Detta innebär att hjulen på den ena sidan av roboten kommer att snurra något snabbare än hjulen på andra sidan vilket kommer få roboten att svänga. Om höger eller vänster motor snurrar snabbast bestäms av åt vilket håll roboten ska svänga. Detta bestäms, som nämnts tidigare, av vilket tecken vinkelåterkopplingen från logikmodulen har.

Till motorerna går en pulsbreddsmodulerad signal, en så kallad PWM-puls. Denna genereras med hjälp av en inbyggd funktion i den Atmega16-processor som används på styrmodulen. Då styrmodulen startas upp sätts Timer/Counter1 till att generera 16-bitars icke inverterade PWM-pulser. Registret TCCR1B sätts sedan så att klockskalningen på PWM-pulsen blir 1. När detta är gjort sätter man de två 8-bitarsregistren OCR1A och OCR1B med den signal som ska läggas ut på motorerna. PWM-pulsen som styr motorerna kommer då ut på pinnarna 18 respektive 19 (OC1B respektive OC1A).

För att bestämma vilken riktning motorerna ska snurra sätts pinnarna 17 (PD3) respektive 20 (PD6) till utportar. Låg signal på pinnarna innebär att motorerna snurrar framåt och hög signal innebär att de snurrar bakåt. Följande portar hör till den motor som styr vänster respektive höger hjulpar.

Hjulpar	PWM port	Riktninsport
Vänster	19 (OC1A)	17 (PD3)
Höger	18 (OC1B)	20 (PD6)

8.3 Backa

Om styrmodulen ges en negativ längdkoordinat ska roboten backa istället för att, som ovan, köra framåt. Den relativa vinkeln från logikmodulen ska då inte längre mätas mellan färdriktningen och riktningen till målet. Nu mäts den istället från robotens "rakt bakåt" (den måste med andra ord korrigeras 180^o jämfört med då roboten kör framåt). En negativ vinkel innebär dock fortfarande att målet ligger till vänster om roboten precis som förut.

Roboten backar endast om den får en negativ längdkoordinat som återkoppling. Om en målpunkt ligger bakom roboten och en positiv längdkoordinat skickas så roterar roboten därmed 180^o för att sedan närma sig målet framifrån.

8.4 Rotera på plats

Om det relativa avståndet till målet är 0 så roterar roboten på stället. Detta görs genom att på samma sätt som förut bestämma en grundhastighet och en styrsignal, som beräknas med PD-regulatorn beskriven ovan. Istället för att addera styrsignalen till grundhastigheten på den ena motorsignalen och subtrahera styrsignalen på den andra så adderas nu styrsignalen till grundhastigheten för båda motorsignalerna. Sedan låter man ena motorn snurra bakåt medan andra snurrar framåt. Hjulparen kommer då få samma hastighet men olika riktning vilket innebär att roboten roterar på stället.

8.5 Parkeringsläge

Styrmodulen kan ta emot ett stoppkommando (se kommunikationsavsnitt). Om ett sådant erhålls sätts alla motorsignaler till konstant 0 och roboten står helt stilla. För att tas ur parkeringsläget måste ett startkommando erhållas.

8.6 Kommunikation

Kommunikationen med de övriga modulerna sker seriellt via kommunikationsmodulen. Styrmodulens kommunikation kommer till övervägande del endast bestå i att ta emot de löpande uppdateringarna av återkopplingen till regulatorn som förutsätts komma med jämna intervall. Vissa andra meddelanden kan dock även mottagas och dessa sammanfattas i nedanstående tabell.

Funktion	**Data**	**Beskrivning**
IDENTIFY	none	Begär att modulen ska identifiera sig med sin adress.
DRIVE_STATE	PARKED	Sätter modulen i parkerat tillstånd.
	NOT_PARKED	Starta modulen från parkerat tillstånd.
P_VALUE	uint8_t	Sätter ett nytt värde på P-konstanten i PD-regulatorn.
D_VALUE	uint8_t	Sätter ett nytt värde på D-konstanten i PD-regulatorn.
R_VALUE	sint8_t	Återkopplingdata, realtivt avstånd.
THETA_VALUE	sint8_t	Återkopplingdata, realtiv vinkel.

8.6.1 Tillgängliga data

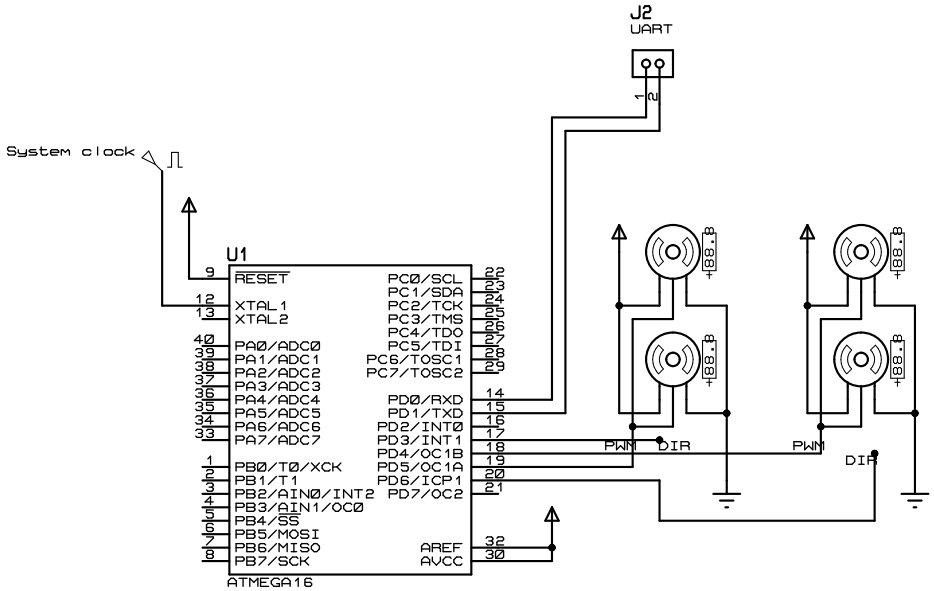
Då någon modul efterfrågar data från styrmodulen sänds denna data uppdaterad kontinuerligt från styrmodulen tills dessa att ett nytt meddelande erhålls som säger att detta inte längre behövs. Vilka data som kan efterfrågas kan ses i tabellen nedan.

Datanamn	**Typ**	**Innehåll**
P_VALUE	uint8_t	P-konstantens värde i PD-regulatorn.
D_VALUE	uint8_t	D-konstantens värde i PD-regulatorn.
DRIVE_STATE	PARKED	Styrmodulen befinner sig i parkerat tillstånd.
	NOT_PARKED	Styrmodulen befinner sig inte i parkerat tillstånd.

En begränsning i kommunikationen är att styrmodulen endast kan hålla reda på att en modul efterfrågar en viss data. Om två olika moduler efterfrågar samma data sänds data endast till den som efterfrågat det senast. Detta är inget problem i dagsläget men kan bli det vid vidareutveckling.

8.7 Kopplingsschema

Nedan i figur 7 visas ett kopplingsschema för styrmodulen.



Figur 7: Kopplingsschema för styrmodulen

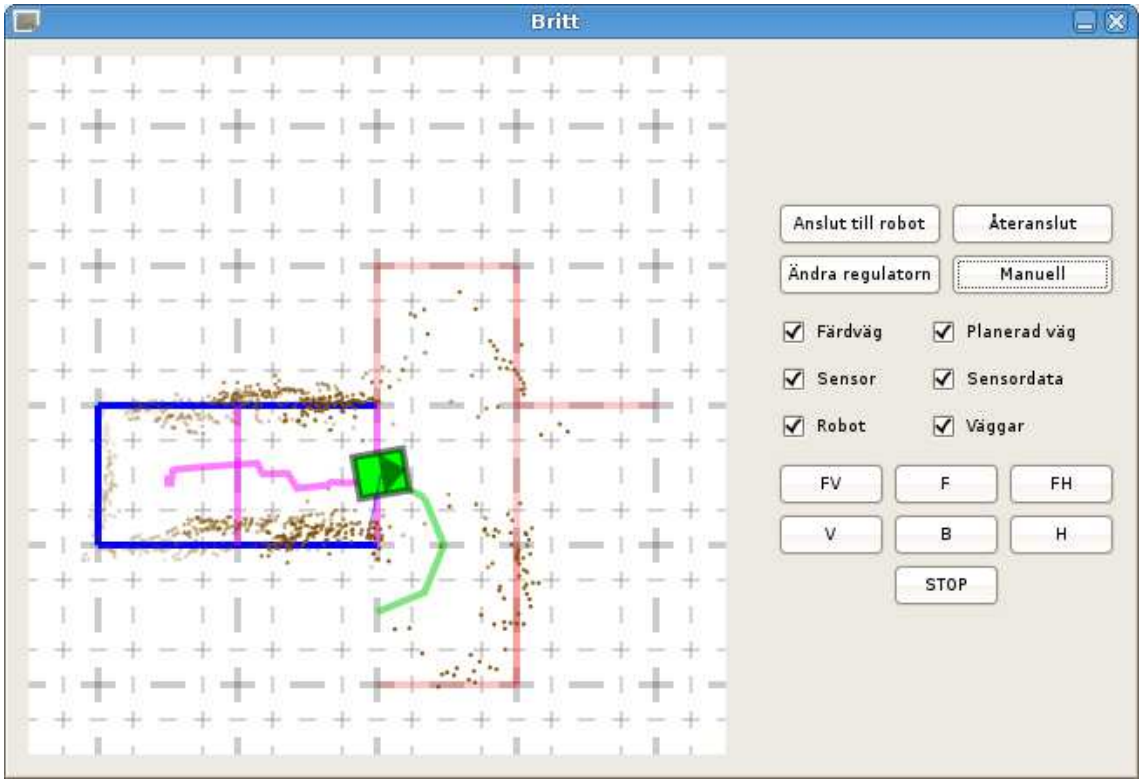
9 Datorprogram

Datorprogrammets huvudsakliga uppgift är att vara ett användargränssnitt till roboten. Efter att användaren startat programmet måste programmet kopplas upp mot roboten. Detta görs genom att klicka på knappen “Anslut till robot”. När programmet är uppkopplat mot roboten står det i manuellt läge.

I detta läge kan användaren via knappar i användargränssnittet styra roboten. När användaren klickar på en av styrknapparna skickas motsvarande styrkommando till roboten, till exempel “Åk framåt” eller “Sväng vänster”. För att underlätta felsökning och finjustering av regulatorn kan man via knappen “Ändra regulatorn” ändra P- och D-parametrarna i en dialogruta.

Genom att klicka på “Autonom”-knappen växlar man roboten till autonomt läge. I detta läge utforskar roboten, efter en inledande kalibrering, den bana som den befinner sig i och skickar data kontinuerligt till datorprogrammet. I datorprogrammet är det möjligt att välja vilka data som ska visas på kartytan genom att kryssa respektive kryssruta. Genom att välja att visa “Färdväg” och “Planerad färdväg” visar ett lila spår var roboten varit och ett grönt vart den planerar att åka härnäst. Kryssrutan “Sensor” väljer att rita upp vilken riktning sensortornet står i som en linje från roboten till senaste mätpunkt. “Sensordata” visar de mätpunkter som roboten fått in från sensorn som bruna prickar. Kryssrutan “Robot” bestämmer om själva roboten ska ritas. Den är representerad av en grön rektangel med en svart pil som visar vilket håll som är framåt. Slutligen väljer “Väggar” om den karta som roboten hittills skapat ska ritas upp. “Manuell”-knappen avbryter autonomt läge och växlar tillbaka till manuell styrning.

I autonomt läge kan datorprogrammet till exempel se ut som i figur 8. Sensormätvärden visas som prickar, planerad färdväg visas som ett grönt streck och den väg som roboten färdats visas som ett lila streck.



Figur 8: Datorprogrammet under autonom körning.

Kommunikationen med roboten sker via TCP/IP med ett dataformat specificerat nedan.

Datafält:	Antal data	Datatyp	Data

där Antal data anger antalet data som följer. Längden anges i bytes.

Nedan följer en lista över datatyper som datorprogrammet kan ta emot.

Datanamn	Typ	Innehåll
MESSAGEQUEUE	kommandolista	en paketerad lista med meddelanden för råa sensordata och annan felsökning
MAP	kartdata	Dump av kartdata
PATH	checkpointdata	Dump av checkpoints
LINE	checkpointdata	Linje från SLAM-korrigering
PING	checkpointdata	Ping-meddelande för att kontrollera att nätverket har ordentlig kontakt.

Mottagna data lagras i matris- respektive vektorform.

Programmet har utvecklats i och för GNU/Linux. Det är skrivet i C++ och använder biblioteket gtkmm for att skapa det grafiska gränssnittet. För kommunikation och trådning används biblioteket boost.

10 Slutsatser

Roboten kan i testmiljö hitta väggar och rita upp dem i datorprogrammet. Det går även att styra roboten från datorprogrammet, och att ändra reglerparametrar, vilket fungerar utan problem, likaså att visa sensordata.

Planeringsalgoritmen klarar av att avgöra om roboten är klar om den får korrekta väggar. Dock kan dåliga mätvärden från IR-sensorerna ge felaktiga väggar. Detta har vi försökt avhjälpa genom att applicera lågpassfiltrering och ha sensorer med olika mätområden. Dessutom räknas endast mätpunkter som träffar mitt på en vägg, inte i hörnen, vilket minskar känsligheten för vinkelfel och brus.

Logikmodulen kan planera för att hitta ej anslutna väggar och lägga upp giltiga rutter. Roboten kör dock ibland fel på grund av positionsavvikelser, eftersom det är svårt att beräkna vinkel exakt från kodhjulen. Detta försöker vi korrigera med hjälp av SLAM-funktioner, som rättar robotens vinkel och position jämfört med väggarna under körning.

Vi har som ett led i utvecklingen också verifierat att den manuella styrningen fungerar.

Referenser

[1] *Designspecifikation. Version 1.2* Hanna Nyqvist. Projektdokument, 2010

[2] *Systemskiss. Version 1.0* Hanna Nyqvist. Projektdokument, 2010

[3] *Kravspecifikation. Version 1.2* Hanna Nyqvist. Projektdokument, 2010

[4] *Projektdirektiv för en kartrobot. Version 1.0.* Thomas Svensson. 2010.

[5] *LIPS – niv 1. Version 1.0.* Tomas Svensson och Christian Krysander. Kompendium, LiTH, 2002.